Rad sa NoSQL bazama podataka

NoSQL je netipičan sistem za upravljanje bazama podataka koji pruža mehanizam za skladištenje i preuzimanje. Dok SQL relacione baze podataka prate koncept odnosa tabela i njihovog povezivanja, NoSQL baza podataka se modelira drugačije. Sam naziv ovog sistema za upravljanje u svom prefiksu sadrži reč *No* kao indikaciju da se relacije u njemu ne koriste.

Ovaj SUBP obuhvata širok izbor različitih tehnologija baza podataka koje su razvijene kao rešenje problema velike količine podataka koje je neophodno skladištiti u aplikacijama, poput korisničkih podataka i objekata, kao i problema učestalosti pristupa tim podacima, brzine pristupa, skaliranja i obrade. Struktura ovih sistema prati više koncepata modeliranja, kao što su parovi ključ–vrednost, grafikoni, podaci orijentisani na dokumenta i podaci orijentisani na kolonu.

Tokom višegodišnjeg razvoja softvera, koristile su se SQL (Structured Query Language) baze podataka zasnovane na modelu povezanih tabela. Poslednjih godina, sa ogromnim porastom korišćenja interneta i *Web 2.0* aplikacija, baze podataka počele su da prevazilaze i veličinu od nekoliko hiljada terabajta. Pojava aplikacija kao što su Google, Amazon, Facebook, Instagram i druge, stvorile su čitavu novu eru upravljanja bazama podataka koja sledi koncept jednostavnog dizajna i velike brzine pristupa i upravaljanja podacima. Ovakve baze podataka pogodne su za aplikacije koje skladište ogromnu količinu podataka, pružaju interakciju u stvarnom vremenu i zahtevaju mogućnost analize podataka kojima raspolažu.

Napomena

Web 2.0, poznat kao participativni ili društveni web, odnosi se na web stranice koje naglašavaju sadržaj koji je generisao korisnik. Karakteriše ih jednostavnost korišćenja, kultura participacije i interoperabilnost za krajnje korisnike. Klasičan primer web 2.0 aplikacija su društvene mreže.

Uzmimo za primer Instagram aplikaciju, koja čuva podatke o svojim korisnicima u bazi. Ako bismo u jednostavnu bazu aplikacije pokušali da uvedemo neku funkcionalnost poput komentarisanja fotografija, korišćenje tipičnog sistema za upravljanje relacionim bazama zahtevalo bi kompletnu reviziju našeg postojećeg dizajna baze podataka. Sa druge strane, NoSQL omogućava laku zamenu ove strukture, budući da definisanje strukture ovih SUBP ne zahteva unapred definisane kolone (atribute), već se podaci mogu direktno upisati u bazu.

NoSQL baze podataka razvijene su kao rešenje sledećih problema relacionih baza:

- strukturirani i uniformni podaci;
- unapred definisana šema podataka u vidu kolona i tipova podataka koji se u njima skladište, što se kosi sa agilnom metodologijom brzog razvoja;
- vertikalan rast zapisa u tabelama svakim dodavanjem podatka, što zahteva dodatni prostor na serveru.

Radi lakšeg razumevanja, u nastavku ćemo prikazati osnovne razlike između ova dva tipa sistema za upravljanje bazama podataka (tabela 7.1).

Karakteristike SQL i NoSQL sistema za upravljanje bazama podataka			
RB	**Karakteristike**	**SQL**	**NoSQL**
1.	model	relacione baze podataka	nerelacione baze podataka
2.	jezik	koristi CRUD operacije za skladištenje podataka u strukturiranoj formi u cilju obavljanja kompleksnih operacija	dinamična šema za nestrukturirane podatke koji se skladište u vidu dokumenata, grafova ili key–value vrednosti ili po kolonama
3.	skalabilnost	vertikalna skalabilnost – proširenje memorije za skladištenje podataka dodavanjem novih memorijskih komponenti	horizontalna skalabilnost – proširenje memorije za skladištenje podataka instalacijom novih servera
4.	implementacija	zasniva se na ACID osobinama: atomičnost, konzistentnost, izolovanost i trajnost	zasniva se na Brewerovoj CAP teoremi, koja obuhvata osobine: konzistentnost, dostupnost, tolerancija grešaka
5.	upotreba	pogodne za izvršavanje kompleksnih upita	pogodne za modeliranje ogromnih baza podataka
6.	programi	otvorenog koda: Postgres, MySQL komercijalni: Oracle, SQLite	isključivo otvorenog koda: MongoDB, BigTable, Redis, RavenDB, Cassandra, Hbase, Neo4j, CouchDB

Tabela 7.1. Poređenje karakteristika SQL i NoSQL sistema za upravljanje bazama podataka

Modeli NoSQL baza podataka

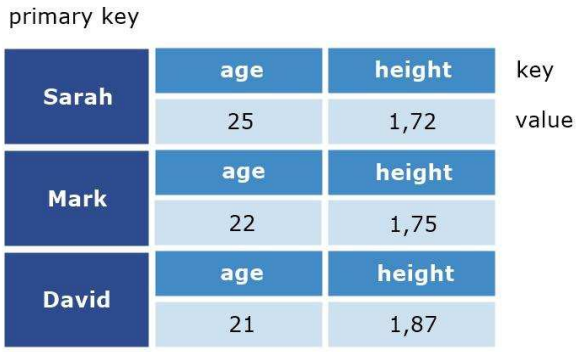
NoSQL baze podataka uglavnom su kategorisane u četiri vrste: model ključ–vrednost, model orijentisan na kolone podataka, model grafa i model orijentisan na dokumenta. Svaka kategorija ima svoje jedinstvene atribute i ograničenja. Nijedna od ovih baza podataka nije bolja od druge za rešavanje svih problema.

Model ključ-vrednost: Ključevi ovako definisanog para su jedinstveni, što omogućava jednostavan pristup podacima. Parovi ključ–vrednost mogu biti različitog tipa: neki čuvaju podatke u glavnoj memoriji, a neki pružaju mogućnost zadržavanja podataka na hard disku. Jednostavan, ali moćan alat koji koristi model parova ključ–vrednost je OracleBerkeleyDB. Još neki koji su često u upotrebi su Riak i Redis.

KEY	VALUE
name	Sarah
age	25
height	1,72

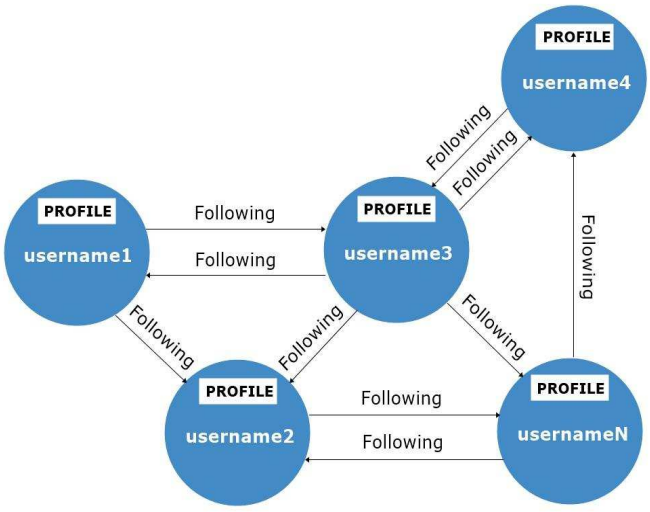
Slika 7.1. Primer modela ključ–vrednost NoSQL baza podataka

Model orijentisan na kolone: Izbegava bespotrebno zauzimanje prostora u bazi. Ako neki podatak ima vrednost *null*, on se u ovom modelu neće skladištiti. Svaka jedinica podataka može se smatrati skupom parova ključ–vrednost, pri čemu se sama jedinica identifikuje pomoću posebnog identifikatora, često nazivanog primarnim ključem. Ovaj primarni ključ se često naziva i ključ reda. Tipičan primer je *Cassandra*.



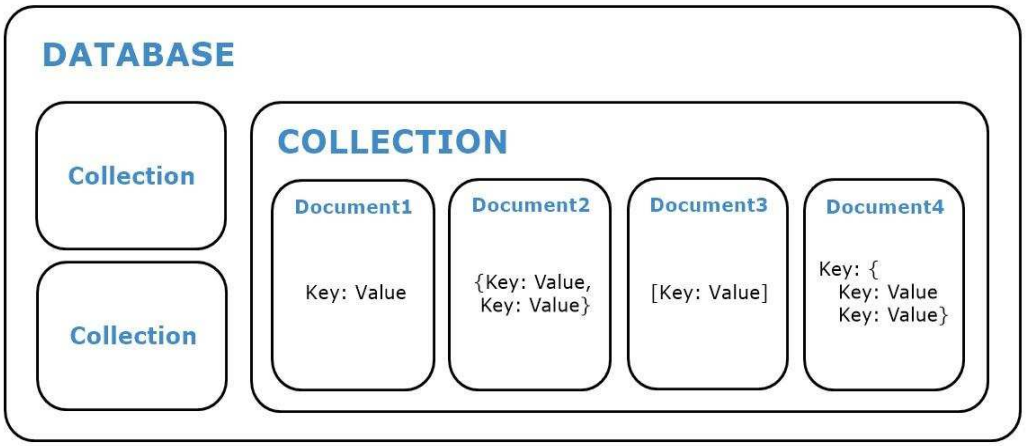
Slika 7.2. Primer modela orijentisanog na kolone NoSQL baza podataka

Model grafa: Baza podataka koristi strukturu grafa sa čvorovima, usmerenim granama koje predstavljaju aktivnosti među tim čvorovima i atributima za opis i čuvanje podataka. Po definiciji, baza podataka grafova je bilo koji sistem za skladištenje koji omogućava povezivanje čvorova bez indeksa. To znači da svaki element sadrži direktni pokazivač na susedni. Opšte baze podataka grafova, u koje se može skladištiti bilo koji graf, razlikuju se od specijalizovanih baza podataka grafova, kao što su baze za trostruko skladištenje vrednosti (subjekat, predikat, objekat) i mrežne baze podataka. Indeksi se koriste samo za pretraživanje grafa. Najpoznatiji model grafa je Neo4J.



Slika 7.3. Model grafa NoSQL baza podataka na primeru društvene mreže Instagram

Model orijentisan na dokument: Tretira dokument kao celinu i izbegava njegovo razdvajanje na parove ključ–vrednost. Na nivou kolekcije, ovo omogućava spajanje raznovrsnog skupa dokumenata u jednu grupu. Baze podataka dokumenata omogućavaju indeksiranje dokumenata ne samo na osnovu njegovog primarnog identifikatora već i na osnovu njegovih atributa. Danas su dostupne različite baze podataka otvorenog koda, a najistaknutije među dostupnim su MongoDB i CouchDB. Zapravo, MongoDB je postala jedna od najpopularnijih ikada korišćenih NoSQL baza podataka.



Slika 7.4. Model podataka orijentisan na dokumente NoSQL baza podataka

MongoDB

Kako sve više podataka postaje dostupno u nestrukturiranom i delimično konstruisanom formatu, povećava se potreba za njihovim upravljanjem pomoću NoSQL baze podataka. Na sličan način kao što komunicira sa relacionim SQL bazama podataka, Python može da komunicira i sa nerelacionim NoSQL bazama podataka.

MongoDB je najzastupljenija NoSQL baza podataka otvorenog koda. Ova baza koristi model orijentisan na dokumente. Dokumenti kojima MongoDB upravlja skladišteni su u JSON formatu. Postoje dva modela podataka koje MongoDB koristi: ugrađeni i normalizovani. Uoči pripreme dokumenta potrebno je odabrati jedan od ta dva modela na osnovu zahteva baze podataka koje želimo da ispunimo.

- **Ugrađeni (denormalizovani) model podataka** pretpostavlja da podatke dobija u razdvojenim dokumentima i teži da ih sve „ugradi" u jedan.
- **Normalizovani model** podatke skladišti u odvojenim dokumentima referencirajući ih na originalni.

Radi lakšeg objašnjenja, u nastavku ćemo dati prikaz ugrađenog i neugrađenog modela na primeru skladištenja podataka o jednom studentu.

Primer ugrađenog modela podataka u MongoDB:

```
{
_id: ,
student_id: "S0000001"
Personal_details:{
      first_name: "Lucia",
      last_name: "Giovanni",
      birth_date: "1997-10-21"
},
Contact_details: {
      e-mail: "lucia.giovanni@gmail.com",
      phone_number: "00124357772368"
}
}
```

Primer normalizovanog modela podataka u MongoDB:

```
#Prikazani objekti se skladište u odvojenim dokumentima

#Student
{
    _id: <ObjectId101>,
    student_id: "S0000001"
}

#Personal_details
{
    _id: <ObjectId102>,
    studentDocID: "ObjectId101",
    first_name: "Lucia",
    last_name: "Giovanni",
    birth_date: "1997-10-21"
}

#Contact_details
{
    _id: <ObjectId103>,
    studentDocID: " ObjectId101",
    e-mail: "lucia.giovanni@gmail.com",
    phone_number: "00124357772368"
}
```

U zadatom primeru prikazani su podaci o studentu. Atributi koje ima student su `student_id`, `Personal_details` i `Contact_details`, koji su u JSON dokumentu predstavljeni u vidu objekata. Atribut **`_id`** predstavlja primarni ključ dokumenta. U MongoDB-u, atribut `_id` osigurava jedinstveni identifikacioni broj kolekcije podataka u dokumentu. Ako dokument izostavi ovaj atribut, MongoDB se stara da njegov objekat automatski bude kreiran. Ovako kreirani objekti poseduju jedinstvene ključeve koji mogu da posluže u referenciranju iz različitih dokumenata, kao što je to prikazano u primeru normalizovanog modela.

Pitanje

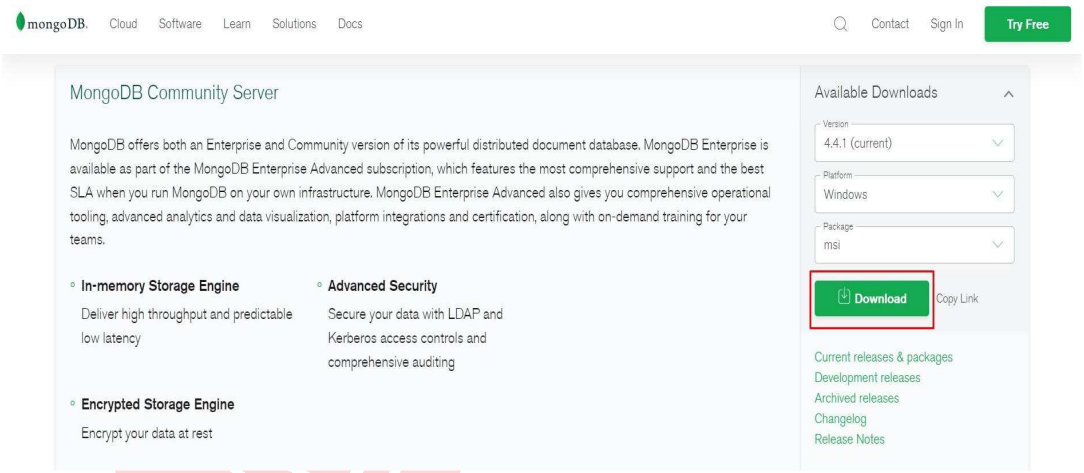
Koja od navedenih NoSQL baza podataka se zasniva na modelu grafa?

- Cassandra
- **Neo4J**
- MongoDB
- Riak
- Redis
- CouchDB

Objašnjenje:

Baza podataka Neo4J se zasniva na modelu grafa, Cassandra na modelu kolona, Riak i Redis na modelu ključ–vrednost, a MongoDB i CouchDB na modelu dokumenata.

Za upotrebu MongoDB-a u Python programu neophodna je instalacija **pymongo** modula, koja se vrši preko komandne linije naredbom **`pip install pymongo`**. Još jedan neophodan korak jeste instalacija MongoDB servera, koju je moguće preuzeti putem zvanične stranice (slika 7.5). Preuzeti fajl je zatim neophodno pokrenuti kako bi instalacija bila izvršena.



Slika 7.5. Preuzimanje instalacije sa zvanične stranice MongoDB

Uputstvo za instalaciju MongoDB-a za Ubuntu:

Potrebno je otvoriti komandni prozor i sledećom komandom uvezli MongoDB javni GPG ključ:

```
wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add –
```

Ova operacija bi trebalo da vrati OK. Međutim, ako dobijete grešku koja ukazuje da gnupg nije instaliran, možete:

Instalirajte gnupg i njegove potrebne biblioteke koristeći sledeću komandu:

```
sudo apt-get install gnupg
```

Kada se instalira, pokušajte ponovo da uvezete ključ:

```
wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add –
```

Kreirajte datoteku /etc/apt/sources.list.d/mongodb-org-5.0.list za svoju verziju Ubuntu-a:

Kreiranje datoteke za Ubuntu 20.04 (Focal):

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

Kreiranje datoteke za Ubuntu 18.04 (Bionic):

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

Kreiranje datoteke za Ubuntu 16.04 (Xenial):

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

Kada ste kreirali datoteku iskoristite sledeću komandu kako biste ponovo učitali lokalnu bazu podataka paketa:

```
sudo apt-get update
```

Kako biste instalirali poslednju verziju MongoDB-a iskoristite komandu:

```
sudo apt-get install -y mongodb-org
```

Da biste pokrenuli i upravljali svojim MongoDB procesom, koristićete ugrađeni init sistem vašeg operativnog sistema. Nedavne verzije Linux-a obično koriste sistemd (koji koristi komandu sistemctl), dok starije verzije Linux-a obično koriste Sistem V init (koja koristi komandu service).

Ako niste sigurni koji init sistem koristi vaša platforma, pokrenite sledeću komandu:

```
ps --no-headers -o comm 1
```

1. Ako koristite systemd MondoDB možete pokrenuti naredbom:

```
sudo systemctl start mongod
```

Ako dobijete grešku sličnu sledećoj kada pokrenete mongod:

```
Failed to start mongod.service: Unit mongod.service not found.
```

Iskoristite sledeću naredbu:

```
sudo systemctl daemon-reload
```

I zatim ponovo pokrenite gornju naredbu za pokretanje.

Možete proveriti da li je MongoDb pravilno instaliran naredbom:

```
sudo systemctl status mongod
```

Opciono možete osigurati da će MongoDB početi nakon ponovnog pokretanja sistema izdavanjem sledeće komande:

```
sudo systemctl enable mongod
```

Po potrebi, možete zaustaviti mongod proces izdavanjem sledeće komande: **`sudo systemctl stop mongod`**

2. Ako koristite system V i init (service) MondoDB možete pokrenuti naredbom:

```
sudo service mongod start
```

Možete proveriti da li je MongoDb pravilno instaliran naredbom:

```
sudo service mongod status
```

Po potrebi, možete zaustaviti mongod proces izdavanjem sledeće komande:

```
sudo service mongod stop
```

Za instalaciju MongoDB-a za različite operativne sisteme i njihove verzije možete pogledati više na stranici: https://www.mongodb.com/docs/manual/administration/install-community/ .

Za korišćenje modula u Python programu potrebno je njegovo uvođenje u program pomoću `import` funkcije. Sledeći korak je konekcija na bazu. Da bismo imali pristup bazi podataka, potrebno je da napravimo instancu klijenta koji će izvršiti konekciju na bazu. Radi jednostavnosti prikaza, koristićemo *test* bazu podataka umesto klasičnog konektovanja na server.

Primer konekcije na *test* MongoDB bazu podataka korišćenjem instance klijenta:

```
from pymongo import MongoClient

client = MongoClient()
db = client.test
```

Umetanje podataka u tabelu u MongoDB vrši se pomoću `insert()` funkcije. Zamislimo da želimo da kreiramo novi kontakt i sačuvamo ga u imenik. Naš imenik predstavlja kolekciju podataka koju kreiramo pozivanjem iz *db* baze podataka pomoću komande `db.contacts`. U okviru promenljive `contacts` definisaćemo podatke o novom kontaktu u JSON formatu, a zatim, korišćenjem `insert_one()` funkcije za unos jednog objekta u bazu, uneti ovaj kontakt.

Primer umetanja podataka korišćenjem `insert()` funkcije u MongoDB-u:

```
from pymongo import MongoClient
client = MongoClient()
db=client.test

contacts = db.contacts
contact_details = {
      'Name': 'Brian Simpson',
      'Phone': '00442561145886'
  }

contacts.insert_one(contact_details)
```

Pomoću pymongo modula napravite bazu fruits koja sadrži ime voća i cenu voća po kg. U bazu upišite podatke za dve voćke pomoću insert_one metode.

Metoda ažuriranja podataka upotrebljava se na identičan način. U ovu svrhu koristimo funkciju `update()`. Funkciji je u okviru parametra neophodno proslediti uslov pod kojim će se ažuriranje izvršiti, kao i nove informacije o objektu koji je predmet ažuriranja. Izmene se u bazi izvršavaju pomoću operatora **`$set`**, koji predstavlja alijas `$addFields` operatora.

Primer ažuriranja podataka korišćenjem `update()` funkcije u MongoDB-u:

```
db.contacts.update_one(
        {"Name":'Brian Simpson'},
        {
        "$set": {
            "Name":"Homer Simpson",
            "Phone":'00442561145800'
        }
        }
    )
```

Ako bismo želeli da izvršimo pregled nekog podatka u tabeli, koristili bismo funkciju pretraživanja. MongoDB u ove svrhe koristi funkciju `find()`. Namena ove funkcije je da vraća vrednost podatka koji zahtevamo po osnovu uslova koji prosleđujemo kao parametar.

Radi vizuelnog prikaza sačuvaćemo ovaj upit u promenljivu `Queryresult`, a zatim ga prikazati na ekranu pomoću funkcije `pprint`, čije je prethodno uvođenje u program neophodno.

Primer prikazivanja podataka korišćenjem `find()` funkcije u MongoDB-u:

```
import pprint from pprint

Queryresult = contacts.find_one({'Name':'Homer Simpson'})

pprint(Queryresult)
```

Ovako izvršen kod prikazaće sledeći rezultat:

```
{'Name': 'Homer Simpson',
 'Phone': '00442561145800',
 '_id': ObjectId('5f4bd767e3e63301dce33200')}
```

Izmenite fruits bazu tako što ćete za jednom voću promeniti i ime i cenu po kg i ispisati podatke baze na standarndi izlaz.

Još jedna od metoda koje MongoDB podržava je brisanje, odnosno `delete()`. Za njeno izvršavanje takođe je neophodno postavljanje uslova u vidu parametra funkcije. Ukoliko bismo nakon brisanja našeg jedinog kontakta u bazi ispisali podatke, rezultat prikaza bi bio *None*.

Primer brisanja podataka korišćenjem `delete()` funkcije u MongoDB-u:

```
db.contacts.delete_one({"Name":'Homer Simpson'})

Queryresult = contacts.find_one({'Name':'Homer Simpson'})

pprint(Queryresult)
```

Izmenite fruits bazu tako što ćete izbrisati jedno voće i ispisati podatke baze na standardni izlaz.

Zadatak za vežbu

Potrebno je omogućiti korisniku da putem komandne linije unese naziv svoje omiljene knjige i njenog žanra. Podatke je potrebno smestiti u MongoDB nerelacionu bazu podataka. Nakon uspešnog unošenja, potrebno je u komandnoj liniji prikazati korisniku sve podatke iz kolekcije.

Rešenje:

```
from pymongo import MongoClient
client = MongoClient()
db=client.test

name = input("Enter book name: ")
genre = input("Enter book genre: ")

books = db.books
book_details = {
    'name': name,
    'genre': genre
}

books.insert_one(book_details)


result = books.find()

for r in result:
    print("Book: {} | Genre: {}".format(r['name'], r['genre']))
```

Objašnjenje:

Prvi korak jeste uključivanje potrebnog pymongo modula, kreiranje Mongo klijenta i odabir baze podataka, što je u našem rešenju *test*. Nakon toga, kreiramo *books* kolekciju i smeštamo je u promenljivu `books`. Zatim, koristeći `input()` funkciju, možemo od korisnika zatražiti unos podataka o knjizi i te podatke sačuvati. Kada su svi podaci popunjeni, kreiramo objekat sa detaljima knjige, koji kasnije prosleđujemo kao ulazni parametar metodi `insert_one`, kojom čuvamo knjigu u *books* kolekciji. Na samom kraju, nad kolekcijom *books* pozivamo metodu `find`, koja vraća sve rezultate. Koristeći `for` petlju, prolazimo kroz pojedinačne rezultate i prikazujemo ih u pogodnom formatu.

Rezime

- NoSQL je netipičan sistem za upravljanje bazama podataka koji pruža mehanizam za skladištenje i preuzimanje na modelima koji nisu zasnovani na odnosu tabela.
- Prednosti NoSQL-a u odnosu na relacioni model su sledeće: upravlja ogromnom količinom podataka, ne zahteva unapred definisanu strukturu baze podataka, ima horizontalnu skalabilnost.
- Modeli koje NoSQL podržava su: model ključ–vrednost, model orijentisan na kolone, model grafa i model zasnovan na dokumentu.
- Model ključ–vrednost za manipulaciju vrednostima koristi ključeve kao jedinstvene identifikatore podataka (OracleBerkleyDB, Riak, Redis).
- Model orijentisan na kolone podrazumeva da se *null* podaci u ovom modelu ne skladište, zbog uštede memorije. Svaki element koji skladišti vrednosti u parovima ključ–vrednost ima i svoj primarni ključ (ključ reda) za jedinstvenu identifikaciju (Cassandra).
- Model grafa koristi strukturu grafa sa čvorovima koji imaju atribute i usmerene grane koje pokazuju aktivnosti među tim čvorovima. Ne zahteva indekse za strukturiranje, ali ih koristi za pretraživanje (Neo4J).
- Model orijentisan na dokument tretira dokument kao celinu i izbegava njegovo razdvajanje na parove ključ–vrednost. Koristi indeksiranje i za dokumente i za atribute (MongoDB, CouchDB).
- MongoDB je najzastupljenija NoSQL baza podataka otvorenog koda, koja koristi model orijentisan na dokumente i skladišti podatke u JSON formatu na dva načina (ugrađeni i normalizovani).
- Za korišćenje MongoDB baze podataka u Pythonu koristi se pymongo biblioteka.
- Za manipulaciju podacima MongoDB koristi CRUD operacije:
    - `insert()` – prosleđuje podatke za umetanje u JSON formatu;
    - `update()` – ažurira podatke na osnovu postavljenog uslova i vrši izmenu u bazi pomoću operatora `$set`;
    - `find()` – prikazuje podatke na osnovu zadatog uslova;
    - `delete()` – briše podatke koji ispunjavaju postavljeni uslov.